J. Parallel Distrib. Comput. 70 (2010) 183–188

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Reevaluating Amdahl's law in the multicore era

Xian-He Sun *, Yong Chen

Computer Science Department, Illinois Institute of Technology, United States

ARTICLE INFO

Article history:
Received 5 December 2008
Received in revised form
22 March 2009
Accepted 9 May 2009
Available online 22 May 2009

Keywords:
Multicore architecture
Scalability
Scalable computing
Memory wall

ABSTRACT

Microprocessor architecture has entered the multicore era. Recently, Hill and Marty presented a pessimistic view of multicore scalability. Their analysis was based on Amdahl's law (i.e. fixed-workload condition) and challenged readers to develop better models. In this study, we analyze multicore scalability under fixed-time and memory-bound conditions and from the data access (memory wall) perspective. We use the same hardware cost model of multicore chips used by Hill and Marty, but achieve very different and more optimistic performance models. These models show that there is no inherent, immovable upper bound on the scalability of multicore architectures. These results complement existing studies and demonstrate that multicore architectures are capable of extensive scalability.

1. Introduction

Multicore architectures integrate multiple processing units into one chip to overcome the physical constraints of unicore architectures, and their exponentially growing power consumption. Multicore architectures provide a more cost-effective chip organization than their unicore counterparts. *Pollack's Rule* [9] states that microprocessor performance increases are roughly proportional to the square root of the increase in complexity. This means that to provide a 40% performance improvement in a unicore design, we need to double the number of logical units. Multicore architectures offer a cost-effective alternative, delivering more computing capability via parallel processing, while consuming less power and board space.

Multicore architectures provide a new dimension to scale up the number of processing elements (cores) and, therefore, the potential computing capacity. While the technology is readily available and some companies are making large-scale multicore processors with hundreds of cores [8,18–20], the leading manufacturers currently provide the smallest number of cores. IBM's Cell processor [3] has eight cores (plus a master core). Sun Microsystems' T2 processor [21] also has eight cores. AMD's mainstream processors – Phenom families [14], announced in 2007, have only four cores. The Dunnington processor [16] announced by Intel in 2008 has six cores. Although Intel does have a roadmap to build a special-purpose 80-core processor by 2011 [17], their approach is conservative and slow moving compared to the roadmaps of smaller companies. These leading manufacturers may share the same pessimistic view of multicore scalability as that expressed theoretically by Hill and Marty [6].

History seems to be repeating itself. Two decades ago, mainstream vendors, worried about the pessimistic implications of Amdahl's law, made parallel machines only with 2 to 8 processors, such as the IBM 7030 Stretch Data Processing System and Cray Y-MP [15]. Contemporary startups of the time such as nCUBE and Thinking Machines made parallel computers with hundreds or thousands of processors. The introduction of the scalable computing concept in 1988 [4], Gustafson's law, changed the course of parallel system designs to large ensemble sizes. The leading manufactures quickly joined in then making massively parallel machines. Today, IBM's Petaflop machine, the Roadrunner, at Los Alamos National Laboratory has 25,200 processors [22]. Sun Microsystems' Ranger supercomputer at the Texas Advanced Computing Center has 15,744 quad-core processors, for example. The emergence and success of these massively parallel systems does not automatically extend to multicore architecture. Hill and Marty [6] recently applied Amdahl's concepts to multicore architectures and, citing hardware design limitations, pessimistically concluded that the future of scalable multicore processors is questionable. Some others follow up with more limitations of multicore scalability based on Amdahl's law [12]. We apply scalable computing principles that emerged in the decades following Amdahl's 1967 work to multicore architectures and the hardware model proposed by Hill and Marty. Our study shows that multicore architectures are fundamentally scalable and not limited by Amdahl's law. In addition to reevaluating the future of multicore scalability, we identify what we believe will ultimately limit the performance of multicore systems: the *memory wall* [13]. While the memory-wall

* Corresponding author.
E-mail addresses: sun@iit.edu (X.-H. Sun), chenyon1@iit.edu (Y. Chen).

0743-7315/$ – see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2009.05.002

problem may never be solved completely, we are optimistic that its effects at scale can be mitigated to some extent. Thus, based on our observations and analyses, and in contrast to the work of Hill and Marty, we are optimistic as to the future of scalability and multicore architectures.

1.1. Fixed-size and scalable computing

The original idea presented by Amdahl [1] was a general observation about the performance improvement limitation of any enhancement, and was later summarized as *Amdahl's law*. Amdahl's law states that if a portion of a computation, f, can be improved by a factor m, and the other portion cannot be improved, then the portion that cannot be improved will quickly dominate the performance, and further improvement of the improvable portion will have little effect. Speedup is defined as sequential execution time over parallel execution time in parallel processing. Let f be the portion of the workload that can be parallelized and m be the number of processors; then the parallel processing speedup implied by Amdahl's law is:

$$\text{Speedup}_{\text{Amdahl}} = \frac{1}{(1-f)+\frac{f}{m}}. \tag{1}$$

When m increases to infinity, the speedup upper bound is: $\lim_{m\to\infty} \text{Speedup}_{\text{Amdahl}} = \frac{1}{1-f}$. Since most applications have a sequential potion that cannot be parallelized, by Amdahl's law, parallel processing is not scalable. For instance, if 90% of an application can be parallelized and 10% cannot, then with 8 to 16 processors, the 10% sequential work will contribute about 50%–80% of the total execution time, and adding more processors for parallel processing will have a diminishing effect.

A tacit assumption in Amdahl's law is that the problem size, or the workload, is fixed to that which runs on the unenhanced system. The speedup emphasizes time reduction of a given problem. Amdahl's law is thus also called the *fixed-size speedup* model [2,7,10,11]. In 1988, Gustafson introduced the concept of scalable computing and the *fixed-time speedup* model [4]. The fixed-time speedup model argues that powerful machines are designed for large problems and problem size should scale up with the increasing of computing capability. For many practical workloads (e.g. real time applications), the problem size scale-up is bounded by the execution time. Thus, the fixed-time speedup is defined as:

$$\text{Speedup}_{\text{FT}} = \frac{\text{Sequential Time of Solving Scaled Workload}}{\text{Parallel Time of Solving Scaled Workload}}. \tag{2}$$

Supposing the original workload, w, and the scaled workload w', finish in the same amount of time with sequential processing and parallel processing with m processors, respectively; and assuming the scale of the workload is in the parallel processing part only; we have $w' = (1-f)w + fmw$. Therefore,

$$\begin{aligned}\text{Speedup}_{\text{FT}} &= \frac{\text{Sequential Time of Solving } w'}{\text{Parallel Time of Solving } w'} \\ &= \frac{\text{Sequential Time of Solving } w'}{\text{Sequential Time of Solving } w} \\ &= \frac{w'}{w} = \frac{(1-f)w + fmw}{w} = (1-f) + mf. \end{aligned} \tag{3}$$

This equation is known as *Gustafson's law* [4]. It states that the fixed-time speedup is a linear function of m if the workload is scaled up to maintain a fixed execution time. Gustafson's law suggests that it is beneficial to build a large-scale parallel system as the speedup can grow linearly with the system size.

Many applications cannot scale up to meet the time bound constraint due to some physical constraints. In practice, the physical constraint is often the memory limitation. With this consideration in mind, Sun and Ni proposed the *memory-bounded speedup* model [10,11]. Let w^* be the scaled workload under a memory space constraint. The memory-bounded speedup is defined as:

$$\text{Speedup}_{\text{MB}} = \frac{\text{Sequential Time of Solving } w^*}{\text{Parallel Time of Solving } w^*}. \tag{4}$$

Assume that each computing node is a processor-memory pair. Increasing the number of processors, then, will increase the memory capacity as well. Let $y = g(x)$ be the function that reflects the parallel workload increase factor as the memory capacity increases m times. That is $w = g(M)$, and $w^* = g(m \cdot M)$, where M is the memory capacity of one node. We have $w^* = g(m \cdot g^{-1}(w))$. Thus memory-bounded speedup is:

$$\text{Speedup}_{\text{MB}} = \frac{(1-f)w + f \cdot g(m \cdot g^{-1}(w))}{(1-f)w + \frac{f \cdot g(m \cdot g^{-1}(w))}{m}}. \tag{5}$$

Eq. (5) looks complicated, but for any power function $g(x) = ax^b$ and for any rational numbers a and b, we have:

$$g(mx) = a(mx)^b = m^b \cdot ax^b = m^b g(x) = \overline{g}(m)g(x)$$

where $\overline{g}(m)$ is the power function with the coefficient as 1. Since many algorithms have a polynomial complexity in terms of computation and memory requirement, and we can always take the highest degree term to represent the complexity of the algorithm, we can simplify Eq. (5) into:

$$\text{Speedup}_{\text{MB}} = \frac{(1-f)w + f \cdot \overline{g}(m)w}{(1-f)w + \frac{f \cdot \overline{g}(m)w}{m}} = \frac{(1-f) + f \cdot \overline{g}(m)}{(1-f) + \frac{f \cdot \overline{g}(m)}{m}}. \tag{6}$$

We provide a quick example to illustrate the calculation of $\overline{g}(m)$ for matrix multiplication. The computation requirement of matrix multiplication is $y = 2N^3$ and the memory requirement is $x = 3N^2$, where N is the dimension of the two $N \times N$ source matrices. Thus:

$$g(x) = 2\left(\sqrt{\frac{x}{3}}\right)^3 = \frac{2}{3^{\frac{3}{2}}}x^{\frac{3}{2}}, \quad \text{and} \quad \overline{g}(x) = x^{\frac{3}{2}}.$$

Therefore, the memory-bounded speedup for matrix multiplication is:

$$\text{Speedup}_{\text{MB}} = \frac{(1-f) + f \cdot \overline{g}(m)}{(1-f) + \frac{f \cdot \overline{g}(m)}{m}} = \frac{(1-f) + f \cdot m^{3/2}}{(1-f) + f \cdot m^{1/2}}. \tag{7}$$

In general, if we assume each element stored in memory will be used at least once, we have $w^* \geq w'$, and the memory-bounded speedup is greater than or equal to the fixed-time speedup.

Eq. (6) is also known as *Sun and Ni's law* [2,7,10,11]. It is a generalization of Amdahl's law and Gustafson's law, where Amdahl's law is a special case with $\overline{g}(m) = 1$, and Gustafson's law is a special case with $\overline{g}(m) = m$. In general, the computational workload increases faster than the memory requirement, thus $\overline{g}(m) > m$ and the memory-bounded speedup model gives a higher speedup than the fixed-size and fixed-time speedup. Memory-bounded speedup is natural for domain decomposition based applications and can be applied at different levels of a memory hierarchy system. It becomes more and more important with increasing awareness of the *memory-wall problem* [13].

1.2. A simple cost model for multicore chips

Hill and Marty [6] give a simple hardware model for multicore chips. This hardware cost model assumes that a multicore chip under study can contain at most *n base core equivalents (BCEs)* and each single BCE implements the baseline core. This assumption

comes from the fact that the microarchitects can only dedicate limited resources on a chip. This cost model also assumes that microarchitects have the technique to create a more powerful core with *perf(r)* sequential performance with r BCEs, where the performance of a single BCE is assumed to be 1. The value of *perf(r)* depends on the actual hardware technique and implementation, but in analysis, it can be an arbitrary function.

1.3. Fixed-size speedup model of multicore

Following Amdahl's law, Hill and Marty [6] conclude that the speedup of a symmetric multicore architecture is:

$$\text{Speedup} = \frac{1}{\frac{1-f}{perf(r)} + \frac{f \cdot r}{perf(r) \cdot n}}. \tag{8}$$

While it is not given in [6], here we provide the deduction of Eq. (8) so it can be better related to the scaled scalability analysis given in Section 2.

According to speedup definition:

$$\text{Speedup} = \frac{\text{Enhanced performance}}{\text{Original performance}} = \frac{T_{\text{Original}}}{T_{\text{Enhanced}}} \tag{9}$$

where the performance is the reciprocal of the execution time. Let us assume that the problem size is w. Thus the original execution time is $T_{\text{Original}} = w/perf(1) = w$, where a single BCE core has a performance of 1 as assumed by Hill and Marty [6]. The new execution time of n-BCE multicore is

$$T_{\text{Enhanced}} = \frac{(1-f)w}{perf(r)} + \frac{fw}{\frac{n}{r} \cdot perf(r)},$$

if we assume these n-BCE resources are built into n/r cores, where each core has a *perf(r)* performance.
Therefore, the speedup is:

$$\text{Speedup} = \frac{\text{Enhanced performance}}{\text{Original performance}} = \frac{w/perf(1)}{\frac{(1-f)w}{perf(r)} + \frac{fwr}{n \cdot perf(r)}}$$

$$= \frac{1}{\frac{1-f}{perf(r)} + \frac{f \cdot r}{perf(r) \cdot n}}$$

perf(r) is a constant for a given design. Let $perf(r) = c$ and $m = n/r$; then Eq. (8) becomes:

$$\text{Speedup} = \frac{c}{(1-f) + \frac{f}{m}}$$

which is the Amdahl's law that invariably results from a fixed-size workload assumption.

By Eq. (8), the scalability of multicore architectures is rather limited. Fig. 1 illustrates the fixed-size speedup of multicore architectures, where c equals 1. The horizontal axis represents the number of cores, scaled from 1 to 256. The vertical axis represents the speedup value. This figure plots the fixed-size speedup results with f ranging from 0.5 to 0.999. As shown clearly from this figure, the fixed-size speedup model (Amdahl's law) illustrates a very limited scalability of a multicore architecture, and the speedup is quickly restricted by the sequential portion of a problem under study. The scalability is acceptable only when the problem is highly parallelizable, such as the improvable portion being over 99.9%.

2. Scalable computing for multicore architectures

We follow the hardware cost model of Hill and Marty's study and analyze the scalability of multicore architectures with the concept of scalable computing. We assume that the multicore architecture under study is a symmetric architecture, and assume each core has its own L1 cache, where the memory bound is the cumulated capacity of the L1 caches. Please notice that the memory-bounded condition can be applied at different levels

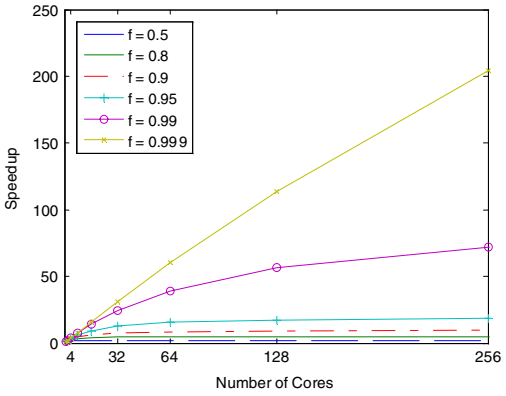


Fig. 1. Fixed-size speedup of a multicore architecture.

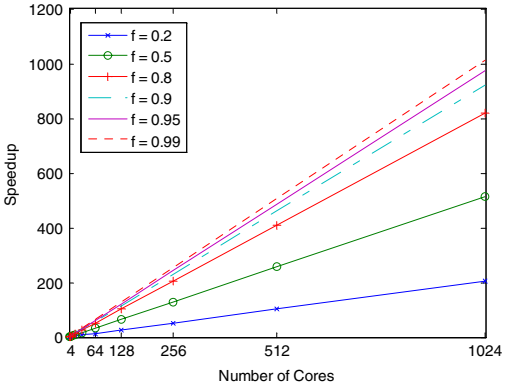


Fig. 2. Fixed-time speedup of a multicore architecture.

of the underlying memory hierarchy. For instance, should the capacity of L2 increases proportionally with the number of cores, the following analyses for L1 can be directly applied to L2.

2.1. Fixed-time speedup model

We take n, the number of base cores, as the scaling factor. The scalability question is whether we should have a large n.

Following Eq. (8), let $n = r$; we have

$$\text{Speedup} = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{perf(r)}} = perf(r).$$

Let $n = r$ be the initial point, and $n = mr$ as the scaled number of cores. Following the fixed-time model assumption that the scaling is only at the parallel portion, for the fixed-time speedup model we have:

$$\frac{(1-f)w}{perf(r)} + \frac{fw}{perf(r)} = \frac{(1-f)w}{perf(r)} + \frac{fw'}{perf(r)m}. \tag{10}$$

Thus, $w' = mw$.

Hence, the scaled speedup, compared with $n = r$ is:

$$\text{Speedup} = \frac{\text{Sequential Time of Solving } w'}{\text{Sequential Time of Solving } w}$$

$$= \frac{\frac{(1-f)w}{perf(r)} + \frac{fw'}{perf(r)}}{\frac{w}{perf(r)}} = (1-f) + mf. \tag{11}$$

Eq. (11) shows that multicore architectures are scalable under the scalable computing model, and their fixed-time speedup grows linearly with the scaling factor.

Fig. 2 reveals the scalability of multicore architectures with the fixed-time speedup model. We compute the speedup following

formula (11) under different scenarios where f ranges from 0.2 to 0.99, and plot the results in Fig. 2. The fixed-time speedup model, as shown in Fig. 2, presents a more optimistic view of the multicore architecture. For instance, when f equals 0.9, the speedup achieved is 922 with 1024 cores, where by Amdahl's law, Eq. (8), the speedup is around 10. When $f = 0.99$, the fixed-time speedup is 1013 with 1024 cores.

The continued performance improvement of fixed-time speedup is due to the fact that it continuously has enough work for parallel processing. After scaling, the parallel work is fw', and the total work is:

$$(1-f)w + fw' = [1 + (m-1)f]w.$$

Thus, the new parallel work over total work ratio is

$$f' = \frac{mfw}{[1+(m-1)f]w} = \frac{f}{\frac{1}{m} + \frac{m-1}{m}f}.$$

When $m \to \infty$, the parallel work ratio approximates to 1. Under the fixed-time model, multicore architectures are scalable and not limited by the sequential processing term.

2.2. Memory-bounded speedup model

Following a similar analysis of fixed-time model, and assuming the scaled workload under memory capacity constraint is w^*, we have the speedup under memory-bounded model, when the number of cores is scaled from r to mr, as:

$$\text{Speedup} = \frac{\text{Sequential Time of Solving } w^*}{\text{Parallel Time of Solving } w^*} = \frac{\frac{(1-f)w}{perf(r)} + \frac{fw^*}{perf(r)}}{\frac{(1-f)w}{perf(r)} + \frac{fw^*}{m \cdot perf(r)}} = \frac{(1-f)w + fw^*}{(1-f)w + \frac{fw^*}{m}}. \quad (12)$$

Assume $y = g(x)$ is the function of computing requirement in terms of memory requirement, $w = g(M)$, and assume $g(x)$ is a power function. Therefore, following Section 1, the memory-bounded speedup is:

$$\text{Speedup} = \frac{(1-f)w + f \cdot \overline{g}(m)w}{(1-f)w + \frac{f \cdot \overline{g}(m)w}{m}} = \frac{(1-f) + f \cdot \overline{g}(m)}{(1-f) + \frac{f \cdot \overline{g}(m)}{m}}. \quad (13)$$

Fig. 3 demonstrates the speedup with the memory-bounded scaled speedup model for multicore architectures. This figure reports the speedup value of the matrix multiplication example (see Section 1). Similar to the fixed-time model, the memory-bounded speedup model reveals that a multicore architecture can scale up well as long as the workload size of the application can be allowed to grow with the number of cores. In addition, the results of the memory-bounded speedup model show that an even better performance can be achieved when the memory capacity constraint is used to scale the workload instead of the execution time constraint. As revealed in Fig. 3, the scalability of a multicore architecture can increase steadily, in contrast with the fixed-time model. The memory-bounded speedup model reflects situations where memory capacity is the constraint, in the case of the L1 cache of multicore architectures as we discussed; the fixed-time speedup model reflects situations where the execution time is limited by human patience or the workflow situation. Both models exhibit a promising view of large-scale multicore architectures.

2.3. Putting it all together

Fig. 4 combines the fixed-size, fixed-time and memory-bounded speedup together for comparison. We pick three scenarios, f with value 0.5, 0.9 and 0.99, for each speedup model. The speedups of fixed-size, fixed-time and memory-bounded models are represented with different line patterns. As illustrated in this figure, with the scalable computing viewpoint and the scaled speedup models, a multicore architecture can scale up well and linearly. The scalable computing notion and models

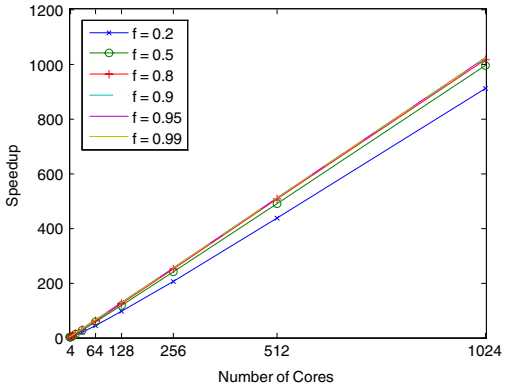


Fig. 3. Memory-bounded speedup of a multicore architecture.

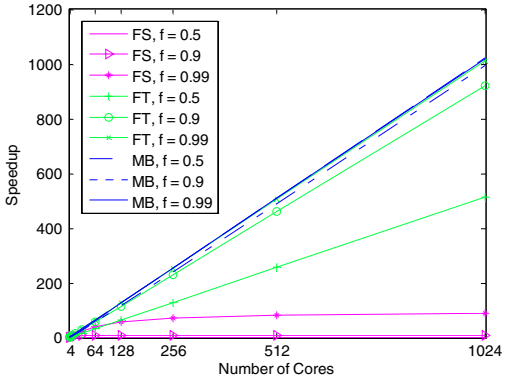


Fig. 4. Fixed-size, fixed-time and memory-bounded speedup of a multicore architecture.

demonstrate a much more optimistic view than Amdahl's law does, and suggest large-scale multicore architectures are of broad value. The direct comparison also verifies that memory-bounded speedup is likely to be higher than fixed-time speedup.

2.4. Results and implications

Result 1. The scalable computing concept and two scaled speedup models, fixed-time speedup model and memory-bounded speedup model, are applicable to multicore architecture design.

Implication 1. Hill and Marty's recent conclusion on the scalability of multicore architecture [6] is essentially a corollary of Amdahl's law. Their analysis and formulation are correct, but, as Amdahl's law, only apply if users do not increase their computing demands when given more computing power.

Implication 2. The scalable computing concept of parallel processing is applicable to multicore architecture. Based on the current success of large-scale parallel computers, multicore architectures with thousands and tens of thousands of cores should be beneficial. Multicore architectures are highly scalable; chip designers need not re-invent the scaling concepts of parallel processing from the last several decades.

Implication 3. Sequential processing is not a limiting factor of multicore scalability, at least not in the sense of scalable computing.

Nonetheless, we are having difficulties utilizing today's multicore systems. A question we have to ask ourselves is: if sequential processing is not the limiting factor for scalability, then what is? We believe the limiting factor is data access delay, or the so-called *memory-wall problem*. Let us revisit the scalability problem considering data access as the factor limiting performance.

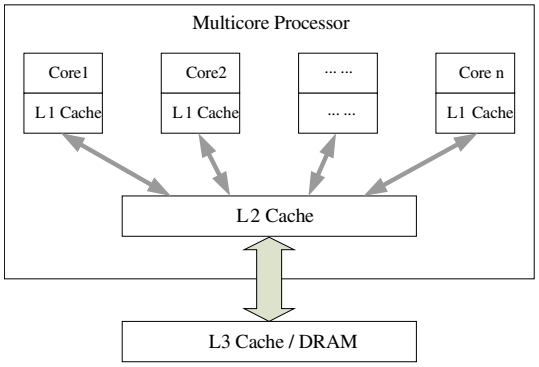


Fig. 5. Multicore architecture.

3. Memory wall and multicore architecture scalability

Multicore processor scalability is not necessarily the same as multicore-processor parallel processing scalability. For many applications, such as meta-tasks, high-throughput computing, or perfectly parallel applications, the sequential portion of the parallel workload is not the limiting factor for performance. Nonetheless, the performance of these types of applications is often limited on multicore architectures.

Fig. 5 illustrates a general structure of a multicore system. The memory-bounded speedup model gives a performance upper bound where all the data are stored in the L1 caches. But, for any actual application with reasonable size, data may have to be accessed through the memory hierarchy, where long data-access delay occurs, a.k.a. the *memory-wall problem* [13], in addition to the contention of the shared L2 cache and data paths to the lower level of the memory hierarchy. The memory-wall problem is due to the disparity of technology advance between CPU speed and memory data access latency. During last three decades memory latency in terms of processor cycles has increased roughly from 0.3 cycles in 1980 to 220 cycles in 2005, and the gap is still growing [5]. In the following, we study the scalability of multicore architecture with data-access delay as the scalability overhead.

For data-access scalability analysis, we have to change the cost model slightly. We assume a task as two parts: data processing work, w_p, and data communication (access) work, w_c, and $w = w_p + w_c$. We assume w_c is a function of r, but it is independent of the workload and the number of cores. As in Section 2, the design choice is to choose an appropriate r to optimize *perf(r)* under the same assumption that the performance of a single BCE core is 1, and the scalability concern is on determining an appropriate number of base cores, n, for best performance. Following a similar deduction as given in Section 1, we have the fixed-size speedup as:

$$\text{Speedup}_{\text{FS}} = \frac{1}{\frac{w_c}{perf(r)} + \frac{w_p \cdot r}{perf(r) \cdot n}}.$$

For fixed-time speedup, taking $n = r$ as the initial point, following the fixed-time principle where $n = mr$, we have:

$$\frac{w_c}{perf(r)} + \frac{w_p}{perf(r)} = \frac{w_c}{perf(r)} + \frac{w'_p}{m \cdot perf(r)}.$$

Thus, $w'_p = mw_p$.

Therefore, the fixed-time speedup compared with r BCEs is:

$$\text{Speedup}_{\text{FT}} = \frac{\frac{w_c}{perf(r)} + \frac{w'_p}{perf(r)}}{\frac{w_c}{perf(r)} + \frac{w_p}{perf(r)}} = \frac{w_c + m \cdot w_p}{w_c + w_p}.$$

If we let $f' = \frac{w_p}{w_c + w_p}$ then we have the familiar format

$$\text{Speedup}_{\text{FT}} = (1 - f') + mf'.$$

For memory-bounded speedup, when the number of cores is scaled from r to mr, we have:

$$\text{Speedup}_{\text{MB}} = \frac{\frac{w_c}{perf(r)} + \frac{f w_p *}{perf(r)}}{\frac{w_c}{perf(r)} + \frac{f w_p *}{m \cdot perf(r)}} = \frac{w_c + f w_p *}{w_c + \frac{f w_p *}{m}} = \frac{w_c + f \cdot g(m \cdot g^{-1}(w_p))}{w_c + \frac{f \cdot g(m \cdot g^{-1}(w_p))}{m}}.$$

For any power function $g(x)$, we have a simplified memory-bounded speedup formula as:

$$\text{Speedup}_{\text{MB}} = \frac{w_c + f \cdot \overline{g}(m) w_p}{w_c + \frac{f \cdot \overline{g}(m) w_p}{m}}.$$

Similar to the analysis in the previous section, it is likely that the memory-bounded speedup is greater than the fixed-time speedup since the computing requirement is generally greater than memory requirement.

Result 2. If we assume the data-access delay is a constant that does not increase with problem size and the number of cores, the scalable computing concept and models are still applicable to multicore architecture.

While the assumption of fixed data-access time is not true under today's technology, it is a technical issue not an inherent, immoveable obstacle.

Implication 4. There is no innate limitation to scalability, but the need for technical improvements is primarily in memory performance.

Since the improvement of memory performance cannot be limited at the microprocessor or cache level only, we have the following implication.

Implication 5. The scalability issues of multicore architecture involve the whole architecture design of a computing system.

The memory-wall problem is a complicated technical issue, yet for the scalability of multicore architectures we only need the data delay to be constant. With research and technology advance, we should be able to mitigate the memory-wall effect and provide a much better performance than that offered by today's multicore architecture.

Implication 6. Scalable computing calls more research to overcome the technical hurdles, especially in reducing the data access delay.

4. Conclusion

We have studied the scalability of multicore architectures. Based on the scalable computing concept where problem size can be increased with the computing power (the number of cores), we have derived two sets of performance model for considering sequential processing and data access (memory wall) as the hampering factors of scalability. We provide three performance models: fixed-size, fixed-time, and memory-bounded, for each hampering factor. The fixed-size performance model with the sequential processing factor is the symmetric model given by Hill and Marty [6] based on Amdahl's law. Unlike the fixed-size model, the *fixed-time* and *memory-bounded* models, where the problem size increases in accordance with the time and memory constraints, scale well. They show that multicore architecture is scalable and has a bright future.

We have only studied symmetric multicore architectures where all the cores are identical. The reason is that asymmetric systems are much more complex than their symmetric counterparts. They are worth exploring only if their symmetric counterparts cannot deliver satisfactory performance. Whereas we have already shown symmetric multicore architectures are scalable, we believe data

access is the real concern for multicore. Asymmetric design may need to be considered to improve data-access speed.

History is repeating itself. Two decades ago we debated the role of Amdahl's law on the scalability of parallel processing. Today, we seem to be repeating the debate on multicore architecture. Amdahl's assumptions limit scalability, which is constrained by the sequential execution time. The fixed-time and memory-bounded performance models show that there is no innate limitation to scalability, but the need for technical improvements is primarily in memory performance. Many technical issues have to be addressed to reach the potential of scalable computing. For instance, we assume the data-access delay is fixed and is independent of the number of cores and problem sizes in our modeling. This assumption is not true under today's technology for most applications.

The scalability of multicore architecture is an area that is largely unexplored. The fixed-size assumption of Amdahl's law is unrealistic and results in pessimistic predictions; this does nothing to encourage healthy growth in the scale of multicore architectures. We hope this study will shatter the pessimistic view of limited scalability of multicore architectures in the industry and academia the way similar views were shattered for parallel processing twenty years ago, and will stimulate a breakthrough in designing large-scale multicore processors. We welcome further discussions in this direction that lead to a better understanding of the scalability of multicore architectures.

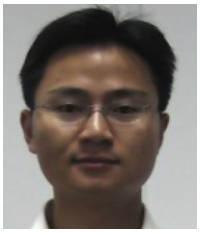
Acknowledgments

We would like to thank Dr. John L. Gustafson, Prof. Kirk Cameron, and Dr. Surendra Byna for their valuable inputs and discussions. This research was supported in part by National Science Foundation under NSF grant CCF0621435, CCF0702737, CNS0751200, CNS0834514, and by United States Department of Energy (DoE) SciDAC program under the contract No. DOE DE-FC02-06 ER41442.

References

[1] G.M. Amdahl, Validity of the single-processor approach to achieving large scale computing capabilities, in: Proc. of AFIPS Conference, 1967.
[2] D.E. Culler, J.P. Singh, A. Gupta, Parallel Computer Architecture: A Hardware/Software Approach, Morgan Kaufmann Publishers, 1999.
[3] M. Gschwind, Chip multiprocessing and the cell broadband engine, Computing Frontiers (2006).
[4] J.L. Gustafson, Reevaluating Amdahl's Law, Communications of the ACM (1988).
[5] J. Hennessy, D. Patterson, Computer Architecture: A Quantitative Approach, 4th ed., Morgan Kaufmann, 2006.
[6] M. Hill, M.R. Marty, Amdahl's law in the multicore era, IEEE Computer 41 (7) (2008) 33–38.
[7] K. Hwang, Z. Xu, Scalable Parallel Computing: Technology, Architecture, Programming, McGraw-Hill, 1998.
[8] J. Makino, K. Hiraki, M. Inaba, GRAPE-DR: 2-Pflops Massively- Parallel Computer with 512-Core, 512-Gflops Processor Chips for Scientific Computing, in: Proc. of ACM/IEEE Supercomputing'07, 2007.
[9] F. Pollack, New Microarchitecture Challenges in the Coming Generations of CMOS Processing Technologies, in: Keynote Speech in the 32nd International Symposium on Microarchitecture, 1999.
[10] X.-H. Sun, L. Ni, Another view on parallel speedup, in: Proc. of IEEE Supercomputing '90, 1990.
[11] X.H. Sun, L. Ni, Scalable problems and memory-bounded speedup, Journal of Parallel and Distributed Computing (1993).
[12] D.-H. Woo, H.-H Lee, Extending amdahl's law for energy–efficient computing in the many-core era, IEEE Computer 41 (12) (2008) 24–31.
[13] W.A. Wulf, S.A. McKee, Hitting the memory wall: Implications of the obvious, Computer Architecture News 23 (1) (1995) 20–24.
[14] AMD website. http://multicore.amd.com/us-en/AMD-Multi-Core.aspx.
[15] Cray Y-MP on Wikipedia. http://en.wikipedia.org/wiki/Cray_Y-MP.
[16] Intel Announces Dunnington Processor. http://www.intel.com/pressroom/archive/releases/20080317fact.htm.
[17] Intel pledges 80 cores in five years http://news.cnet.com/2100-1006_3-6119618.html.
[18] Kilocore Overview. http://www.rapportincorporated.com/kilocore/kilocore_overview.html.
[19] nVIDIA Tesla C870. http://www.nvidia.com/object/tesla_c870.html.
[20] nVIDIA Quadro FX 3700M. http://www.nvidia.com/object/product_quadro_fx_3700_m_us.html.
[21] Sun UltraSPARC T2 Processor. http://www.sun.com/processors/UltraSPARC-T2/.
[22] TOP500 Supercomputing Site. http://www.top500.org.

Xian-He Sun is a Professor of Computer Science and the director of the Scalable Computing Software laboratory at Illinois Institute of Technology (IIT), and is a guest faculty in the Mathematics and Computer Science Division and Computing Division at the Argonne and Fermi National Laboratory, respectively. Before joining IIT, he worked at DoE Ames National Laboratory, at ICASE, NASA Langley Research Center, and at Louisiana State University, Baton Rouge. Dr. Sun's research interests include parallel and distributed processing, software systems, performance evaluation, and data intensive computing. More information about Dr. Sun can be found at http://www.cs.iit.edu/~sun/.



Yong Chen received his B.E. degree in Computer Engineering in 2000 and M.S. degree in Computer Science in 2003, both from University of Science and Technology of China. He is currently pursuing his Ph.D. degree in Computer Science from Illinois Institute of Technology. His research focuses on parallel and distributed computing and computer architecture in general, and on optimizing data-access performance, parallel I/O, performance modeling and evaluation in particular. More information about him can be found at http://www.iit.edu/~chenyon1.